(19) **FRENCH REPUBLIC**

NATIONAL INSTITUTE OF INDUSTRIAL PROPERTY

PARIS

(11) **Publication No.:** **2 778 046**
(To be used only in ordering copies)

(21) **National Registration No.** **98 05110**

(51) Int. Cl.[6] **H 04 L 12/28**, H 04 L 12/24, H 04 N 7/00

(12) **PATENT APPLICATION** **A1**

(22) **Filing date:** April 23, 1998

(30) **Priority:**

(43) **Date application laid open to the public:** October 29, 1999, Bulletin 99/43.

(56) **List of documents cited in preliminary search report:** *See end of this reprint.*

(60) **References to other related national documents:**

(71) **Applicant(s):** THOMSON MULTIMEDIA Société anonyme -- FR.

(72) **Inventor(s):** BICHOT, GUILLAUME; STRAUB, GILLES; COEZ, FABIENNE; and PIRAT, PATRICK.

(73) **Proprietor(s):**

(74) **Agent:** THOMSON MULTIMEDIA.

(54) METHOD FOR MANAGING OBJECTS IN A COMMUNICATIONS NETWORK AND IMPLEMENTING DEVICE

(57) The invention is directed to a method for distributed management of an object catalog in a communication network comprising devices.

The method of the invention comprises the steps of:

- registering local objects that are present in a device in a local register managed at the level of that device,
- propagating an object list request originated by a local object to non-local registers managed by other devices in the network,
- collecting the responses of the non-local registers and the response of the local register and
- transmitting the collected responses to the object that originated the request.

The invention is also directed to devices suitable for being connected to a network in which the above method is used.

The invention finds application particularly in the field of home communication networks.

The invention concerns a method for managing objects, particularly software modules, in a communication network that can be a home network. It also concerns devices suitable for being connected to such a network and comprising means of implementing the method.

The invention finds application particularly in a home network adapted for interconnecting audio and video devices.

In a network of consumer electronics devices such as television sets, cable or satellite decoders or videocassette recorders, it is necessary to provide means of communication between the devices while at the same allowing for the complexity and price constraints inherent in mass-produced devices.

Depending on the type of network contemplated, it may be necessary for a device (also referred to as a "node" hereinafter) in the network to know the access path or address of another device. This is also true if the concept of a device is replaced with that of an object or a software module, a device being able to contain a large number of objects. The elements concerned can be downloaded or resident applications, specific user interfaces or lower-level modules. Each object or module of a device can seek to communicate with another object or module of the same device or of another device in the network. Each object is considered to be a resource available to other objects.

The problem that arises in this case is how to obtain a, or the, list of resources available in the network.

In this context, the invention is directed to a method for distributed management of an object catalog in a communication network comprising devices, characterized in that it comprises the steps of:

- registering local objects that are present in a device in a local register managed at the level of that device,
- propagating an object list request originated by a local object to non-local registers managed by other devices in the network,
- collecting the responses of the non-local registers and the response of the local register and

- transmitting the collected responses to the object that originated the request.

Each node (or device) stores only the data local to that node or device; the data are not duplicated in other nodes. The memory needs in each device are thereby limited.

The search for objects (software modules) is distributed over plural devices, each database being queried at the local level. Information processing resource needs at the local level are also limited in this way.

Data consistency is maintained in a simple manner: it is unnecessary to update remote databases by means of complex processing operations when a change is made in a local database.

If a node disappears, only the data relating to that node are lost.

According to a particular embodiment, an object has an address (SEID) composed of an identifier of a device in which it is present, said identifier being unique in the network, plus a local identifier unique to that object in said device.

According to a particular embodiment, a local register includes, for each object registered in it, the address of that object in the network and the attributes of said object.

According to a particular embodiment, the step of propagating a request comprises the step of determining which devices connected to the network have their own local registers and transmitting the request to the registers of these devices.

An object thus can originate a request to obtain a list of other objects without concerning itself with whether or not these objects are located in the same node.

According to a variant embodiment, a request is propagated only to a given set of remote registers.

For example, when it is known in advance that software modules having certain attributes are present only in a given type of device, it is then possible to limit the propagation of requests to that type of device in order to limit the number of messages circulating in the network.

The invention is also directed to a device in a communication network, comprising:

- means of storing (3, 4) local objects (21 to 28),
- means of storing (3) a local register (26) suitable for containing a local object catalog,
- means (31) of connecting to the network,
- means of transmitting messages from a local object to another object and of propagating an object list request to non-local registers.

According to a particular embodiment, each object has an address that is unique in the network (SEID), the purpose of an object list request being to establish a list of object addresses that meet given criteria.

According to a particular embodiment, the communication network comprises an IEEE 1394 bus.

Other characteristics and advantages of the invention will emerge from the description of a non-limiting, particular embodiment illustrated in the annexed figures, wherein:

- Fig. 1a schematically depicts a home network comprising four devices of different types;
- Fig. 1b is a block diagram of one of the devices of Fig. 1;
- Fig. 2 is a diagram illustrating the software organization of the device of Fig. 1b;
- Fig. 3 shows the states of a register module of a network device,
- Fig. 4 shows the sequencing of messages in connection with a request that is to be propagated to a remote register module.

According to the present embodiment, the home network comprises four types of devices: full-function audio/video devices (FAV), intermediate-function audio/video devices (IAV), base audio/video devices (BAV) and legacy audio/video devices (LAV). The communication bus is an IEEE 1394 bus, but it can be of another type. The network accepts a common command language known as the HAVI language.

The FAV devices have the most complete functionalities of all the devices in the network: a communication manager, a register module, a device control module manager and device control modules (referred to hereinafter as "DCMs"), the latter being downloadable. According to a variant, the device also comprises a user interface manager. The FAV devices can take control of less sophisticated devices, such as BAV and LAV devices, by means of device control modules. An FAV device can access other FAV or IAV devices in order to access resources it does not have (a user interface manager, for example).

The IAV devices have the same functionalities as an FAV except for the ability to download device control modules.

The BAV devices have a private command language that is specific to them and is not necessarily used by the rest of the devices. This type of device is controlled by an FAV device via a control module (DCM) that is downloaded from the BAV device itself and is adapted to control it. For example, the BAV device can be a printer whose print manager is downloaded.

The LAV devices are devices connected to the bus or to IAV or FAV devices via specific connections. The LAV devices are controlled by specific control modules (DCMs) that do not emanate from the device itself and have their own private language.

Figure 1a shows an example of a network in the form of four devices, FAV, IAV, BAV and LAV. The FAV, IAV and BAV devices are connected to the same bus, while

the LAV device is connected directly to the FAV device and controlled by a control module present therein. The BAV device is controlled, for example, via the IAV device.

Figure 1b is a schematic diagram of the FAV device 1. It comprises a microprocessor 2 connected to a read/write memory 3 and an at least partially programmable read-only memory 4, as well as a 1394 interface (reference numeral 5) constituted by an interchange circuit and a physical circuit. Device 1 also comprises a specific interface 6 for connecting the videocassette recorder LAV.

Four types of software modules in particular can be present in the memories of the devices in the present network. These are device control modules DCM, applications, service modules and a message transmission manager.

The device control modules DCM serve to control a device or a subassembly of that device. The control module can be located in the to-be-controlled device itself (if the latter is an IAV or FAV device) or in a device other than the device to be controlled (if the device to be controlled is an LAV or BAV, the control manager will be located in an IAV or FAV device, the latter serving as an execution platform). A control manager is either present from the outset or can be downloaded. In the latter case, the download takes place during the initialization of the device or at the request of an application.

The function control modules (called FCMs) are software modules that serve to control a function of a device and are included in the device control modules DCM. One such device can have plural functions: registration, tuning, camera, display, mass storage, etc.

The service modules offer system functions or services. They can be accessed either by local software modules or via the system that transmits messages to modules of other devices. These system functions or services include in particular user-interface graphics management, management (for example downloading) of the DCM modules, the procedures for connecting a device to the network, initialization of the network (listing of network resources), as well as the register module, which will be seen in

greater detail later on.

Each software module (DCM or application or system services modules) must register with the local register module (that is, the register module in which it resides or into which it has been loaded) if it wants to provide access to other software modules of the network via the message layer.

The message transmission managers are responsible for communicating messages from one software module to another, regardless of the devices in which these modules are located. When a software module wants to send a message to another module, it does not know what physical device houses the recipient module.

Figure 2 illustrates an example of the software organization of an FAV device.

This device comprises a downloaded application 21 (for example a game), two control modules DCM A and B 22 and 23, a private application 24 (for example an electronic program guide), an IEEE 1883/1394 communication bus manager 25, a register module 26, a high-level user interface 27, a DCM manager 28 and a message transmission system 29. The modules communicate with one another via the message transmission system, which can be accessed via an application programming interface 30 (referred to hereinafter as an "API"). The device also comprises an interface 31 with the 1394 bus.

On being installed in the network, the FAV device will seek to load the control modules DCM of the BAV devices in order to make them available for its applications. To this end, each of the BAV devices places the code of the appropriate DCM module in a known area of its own memory and in a self-descriptive data structure known as an SDD (Self-Describing Device). The FAV device can thus come in and read this memory space and load the DCM module of the BAV device. One example is that of the BAV device being a printer. The DCMs loaded in this way are registered in the register module of the FAV device and can thus be accessed by the rest of the network.

An SDD data structure is imperative in FAV, IAV or BAV devices, and is located at a fixed address in each device. It thus becomes possible for an FAV device, during

its initialization, to scan the network in order to load the DCM modules of all the BAV devices. This task is performed by the DCM module manager of the FAV device.

An SDD data structure also includes the type of device (FAV, IAV, BAV, LAV).

The message transmission system of a device comprises:

- the register module 26 (declaration of and search for software modules),
- a message layer, comprising the message dispatcher (message transmission and reception), the application programming interface 30 (API), for accessing the transmission system, and an IEEE 1394 bus adaptation sublayer.

The IEEE 1394 bus adaptation sublayer serves primarily to adapt data transmission to the IEEE 1394 protocol by encapsulating the messages to be transmitted into packets adapted to the IEEE 1394 standard.

A message includes three components: the address of the recipient software module, the address of the source software module and useful data.

The address of a software module is composed of an identifier of the node on which it is executed, this identifier being unique in the network, followed by a software module identifier unique to the node in which it is executed. Software module identifiers are allocated by the message transmission system. The addresses are used by the message dispatcher to send messages to the appropriate software module. According to the present embodiment example, a software module address or identifier (referred to hereinafter as an "SEID") is an 80-bit binary word. It comprises:

- a 64-bit device identifier, stored in the ROM of the device in which the software module is executed. The host's identifier is used for downloaded modules. The device identifier is assigned at manufacture and corresponds to field EU164 defined in IEEE Standard 1394-1995. Part of this device identifier is administered by the IEEE organization and is specific to each manufacturer; the other part is chosen by the device manufacturer itself in such a way that each manufactured device is given a different identifier;
- a local identifier constituted by a serial number assigned directly by the

message transmission system of a node, this number being coded on 16 bits and concatenated with the device identifier to form the SEID identifier. The message transmission system maintains a counter for this purpose. A certain number of serial numbers (for example from 0x0000 to 0x0005) are set aside to identify particular service modules. For example, the serial number 0x0001 systematically corresponds to the register module of a device.

Hence, every software module in the network has a distinct, unique SEID identifier. It is possible, however, to define unique identifiers by means other than those indicated above.

The register module maintains a database that includes a directory of the software modules available over the network. It provides a programming interface that affords access to the functions of software module registration and performing module searches based on a list of criteria.

There is one register module in each FAV or IAV device. Within such a device, all the software modules are registered by the local register module. If a software module wants to be contacted, it must register with the register module. The register module maintains the network address and the attributes of every software module registered with it.

According to a variant embodiment, the register module includes the serial number of the software module instead of its address.

The attributes of a software module serve to characterize it. For each software module, these attributes are stored in a table that gives for each attribute the 32-bit reference of the attribute, its size in bytes, and its value.

Table 1 gives a list of predetermined attributes:

Attribute Reference	**Type Format**	**Size**	**Presence**
Type of software module	Integer	32 bits	M
HUID identifier	Byte string	80 bits	M*
Type of device	Integer	32 bits	M*

Graphic interface	Integer	32 bits	O
Medium format	Bit field	32 bits	O
Data format	Bit field	32 bits	O
Manufacturer of device	Character string	15 bytes	M*
Manufacturer of software module	Character string	15 bytes	O
Version of software module	Character string	15 bytes	O
Audio/video command language	Bit field	32 bits	O

Table 1

The type of software module represents the primary function of the module. If the software module is a system service module, then the type of the attribute denotes the system service itself. The register module is a service module of this kind. If the software module is a function control manager FCM, the type defines the function: registrator, display, tuner, etc. If the software module is a device control manager DCM, the type is "DCM." If the software module has an application programming interface ("API") that is not compatible with the rest of the network, then the type is "private."

The "HUID identifier" is an identifier of a device with which a DCM manager is associated or a function with which an FCM manager is associated.

The type of device associated with the software module is FAV, IAV, BAV or LAV, as already explained.

A DCM manager can be associated with a graphic user interface. The attribute "Graphic interface" indicates whether this is the case, and if so, the degree of compatibility of the DCM manager's interface with the various interface levels provided in the network.

The attribute "Medium format" indicates the type of data storage medium

supported by a device. This can be, for example, a DVD, DAT, DVHS or DVC medium.

The attribute "Data format" indicates the data format that can be manipulated by a device. This can be, for example, the MPEG2, JPEG, MIDI or ASCII format.

The attributes "Manufacturer of device" and "Manufacturer of software module" indicate a reference for the manufacturer of the device or of the software module, respectively, while the attribute "Version of software module" indicates the version number of a module.

Finally, the attribute "Audio/video command language" indicates the types of languages specific to the software module in addition to the common HAVI command language mentioned above. The value of the attribute is a 32-bit field, with the value of each bit indicating compatibility with a specific command language.

In a variant embodiment, the database of a register module can also include specific or "private" attributes.

It should be noted that the register modules of different devices are distinct. There is no centralized register in which all the software modules are listed.

According to the present embodiment, the application programming interface of a register module includes five commands, which will be described in detail below:

(a) Register a software module

This command is used to add a software module to the database of the local register or to modify the attributes of a software module that has already been registered. It is used primarily by a software module to register itself on connection of the device comprising that element.

The software module transmits its SEID identifier and the attributes to the register module. If this identifier is already present, the new attributes replace the old ones. If not, a new entry is created in the local database (local register). The register module transmits a status message to the software module which will be either a write

confirmation or an error message, depending on the outcome of the write.

(b) Retrieve a software module

This command is used to read the attributes of a software module when its SEID identifier is known. A pointer to an area of the device's read/write memory into which the data are to be copied is transmitted along with the retrieval request. If the software module is not present in the local database, then the pointer is reset to zero and returned by the register module.

The register module also returns a status message that confirms the copying of the attributes or indicates that the sought identifier is not present.

(c) Delete a software module from the register

This command is used to remove a software module from the local database. Its SEID identifier is supplied as a parameter of the command. The register module returns a status message confirming deletion or indicating that the software module in question was not found.

(d) Request list of software modules ("single request")

This command is used to determine the identifiers of the software modules registered in a local database and meeting certain criteria. In the present embodiment example, these criteria are the reference of an attribute and the value of an attribute. A parameter of the command is also an operator indicating the manner in which the comparison of the value of the attribute specified in the command to the values of the base should be performed (equal to, greater than, greater than or equal to, less than, less than or equal to, different, bit-by-bit "AND" logic, bit-by-bit "OR" logic, etc.).

The register module returns the list of SEID identifiers of any corresponding software modules. It also returns a status message indicating the success of the operation (whether any identifiers were found or not) or its failure.

(e) Perform a Boolean operation between two lists of software modules ("multiple request)

This command is used to perform a Boolean operation on two lists of identifiers. The command includes as parameters the requests corresponding to each list. A request can be constituted by the criteria cited above in Paragraph (d) (single request), or by another multiple request.

A parameter of this command is also the Boolean operator to be considered ("AND" or "OR" in the context of the present embodiment).

The register module returns the list of any SEID identifiers and an information status message on the success of the operation or its failure for any reason, such as a lack of resources, for example.

To access other software modules, a software module must know the SEID identifiers of its counterparts. This is not a problem for software modules that are registered in the same register module, since the requests described above enable each software module to retrieve lists of identifiers from the local database.

A software module accesses the local register module via the local message transmission system. It can also access a remote register module and thereby download the identifiers of modules registered with other register modules. To do this, each register module propagates a request that was locally transmitted to it to the register modules of all the other devices. According to the present embodiment example, a remote register module for which no response has been received within a given time interval is ignored.

Each register module receiving the request of the initial register module searches its local database itself and sends any identifiers meeting the criteria of the request back
to the initial register module. The latter then transmits the concatenated list of all the received identifiers to the software module that originated the request. The software module that originated the request can then communicate with the software modules of other devices and use the resources appertaining to them.

Figure 3 is a state diagram of a register module of a device. This diagram includes two states, A and B. State A is the state of waiting for a request from a software module. State B is the state of waiting for a response from the local register module to a request originating in remote register modules.

Table 2 gives the events triggering actions on the part of the local register module and the corresponding departure and arrival states. The references for the events are the same as in Fig. 4.

Event	Meaning	Action	Departure State	Arrival State
41	Registration or Retrieval or Deletion of a software module in/from the base	Determine and transmit response	A	A
42	"Single" or "multiple" list request received from a remote register module	Determine and transmit response	A	A
43	"Single" or "multiple" list request received from a local software module	Determine response for local base and transmit request to remote register modules	A	B
44	All responses have been received.	Determine and transmit final response to local software module.	B	A

Request propagation is performed by the register module. It will be recalled that the identifier of a register module is composed of a manufacturer identifier (set by the IEEE), a device identifier (set by the manufacturer) and a register module identifier, the latter being the same for all the register modules.

To be able to propagate a request, the register module of a device polls all of the devices in the network and obtains their identifiers. It then determines which of these devices also include a register. In the present embodiment, this is only the FAV or IAV devices. Knowing the identifiers of the devices that can be accessed by the network, the register module reads the type of each device from the data structure SDD mentioned above. In this way it eliminates BAV devices. Concatenating each device identifier with the fixed local identifier (serial number) common to all the register modules to obtain [syntax sic] a list of the complete addresses SEID of all the register

modules. A register module obtains the list of device identifiers via the local bus management module (known as the "CMM"), which monitors the connection and disconnection of the devices in the network. This module reads the list of all the nodes connected to the network from a register called "TOPOLOGY_MAP" defined by IEEE Document 1394-1995, Section 8.3.2.4.1. This register is located in a bus managing device (called a "bus manager" in English in the above-cited IEEE document), which keeps the register updated relative to the topology of the network. The address of this device is known by the other devices via means also described in the IEEE document.

Figure 4 is a diagram indicating the sequencing of messages when a request issued by a software module A of a first device is to be propagated to the register module of a second device, a software module B being registered in the register of that second device.

According to the embodiment presented hereinabove, a request issued by a software module to determine all of the non-local software modules is propagated to all the remote register modules. According to a variant embodiment, this type of request can also be limited to one set of remote register modules, for example those of a specific type of device.

Claims

1. A method for distributed management of an object catalog in a communication network comprising devices, characterized in that it comprises the steps of:

- registering local objects present in a device in a local register managed at the level of that device,
- propagating an object list request originated by a local object to non-local registers managed by other devices in the network,
- collecting the responses of the non-local registers and the response of the local register and
- transmitting the collected responses to the object that originated the request.

2. The method as set forth in claim 1, characterized in that each object has an address (SEID) composed of an identifier of a device in which it is present, said identifier being unique in the network, and a local identifier unique to that object in said device.

3. The method as set forth in either of claims 1 and 2, characterized in that a local register includes, for each object registered therein, the address of that object in the network and the attributes of said object.

4. The method as set forth in one of the preceding claims, characterized in that the step of propagating a request comprises the step of determining which devices connected to the network have a local register themselves and transmitting the request to the registers of those devices.

5. The method as set forth in one of the preceding claims, characterized in that a request is propagated only to a given set of remote registers.

6. A device in a communication network, comprising:

- means of storing (3, 4) local objects (21 to 28),

- means of storing (3) a local register (26) suitable for containing a local object catalog,

- means (31) of connecting to the network,

- means of transmitting messages from a local object to another object and of propagating an object list request to non-local registers.

7. The device as set forth in claim 6, characterized in that each object has an address that is unique in the network (SEID), the purpose of an object list request being to establish a list of object addresses meeting given criteria.

8. The device as set forth in either of claims 6 and 7, characterized in that the communication network comprises an IEEE 1394 bus.

KEYS TO FIGURES:

Fig. 1a:

FAV = digital TV
IAV = decoder
LAV = VCR

Fig. 2:

Using the reference numerals already printed in the diagram:

21. Downloaded application
24. Private application
25. Bus manager (CMM)
26. Register module
27. High-level user interface manager
28. DCM manager
29. Message dispatcher
31. 1883/1394 serial bus
...and to the right:
    a. Software modules

Fig. 4.

a. Software module A
b. Register, Device 1
c. Message transmission system, Device 1
d. Registration
e. Single/multiple request
f. Devices?
g. List of identifiers
h. CMM, Device 1
i. List of SEID identifiers
j. Message (Device 2, Device 1, Request)
k. Message (Device 1, Device 2, Request)
l. Message transmission system, Device 2
m. Register, Device 2
n. Software module B

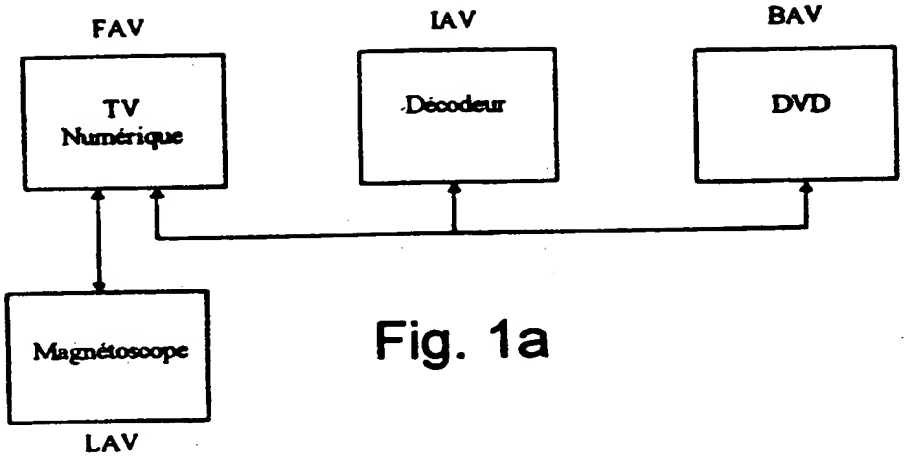
FAV
IAV
BAV
TV Numérique
Décodeur
DVD
Magnétoscope
LAV

Fig. 1a

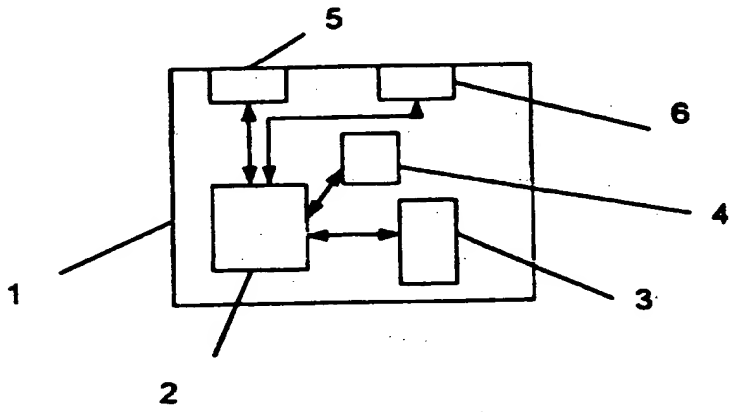
5
6
4
1
3
2

Fig. 1b

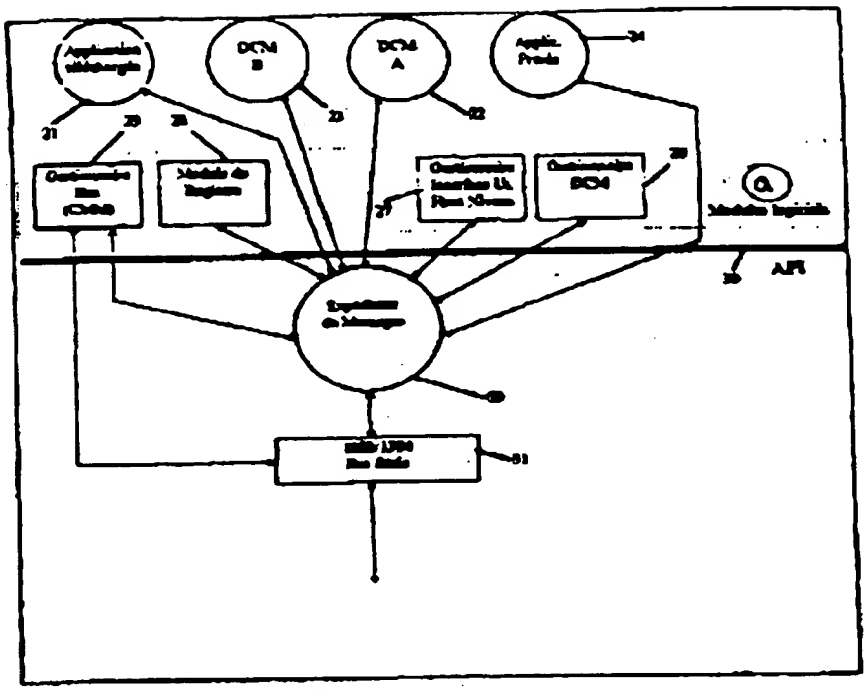

Fig. 2

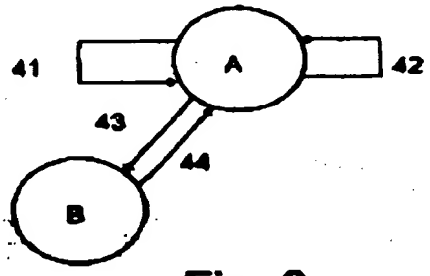

Fig. 3

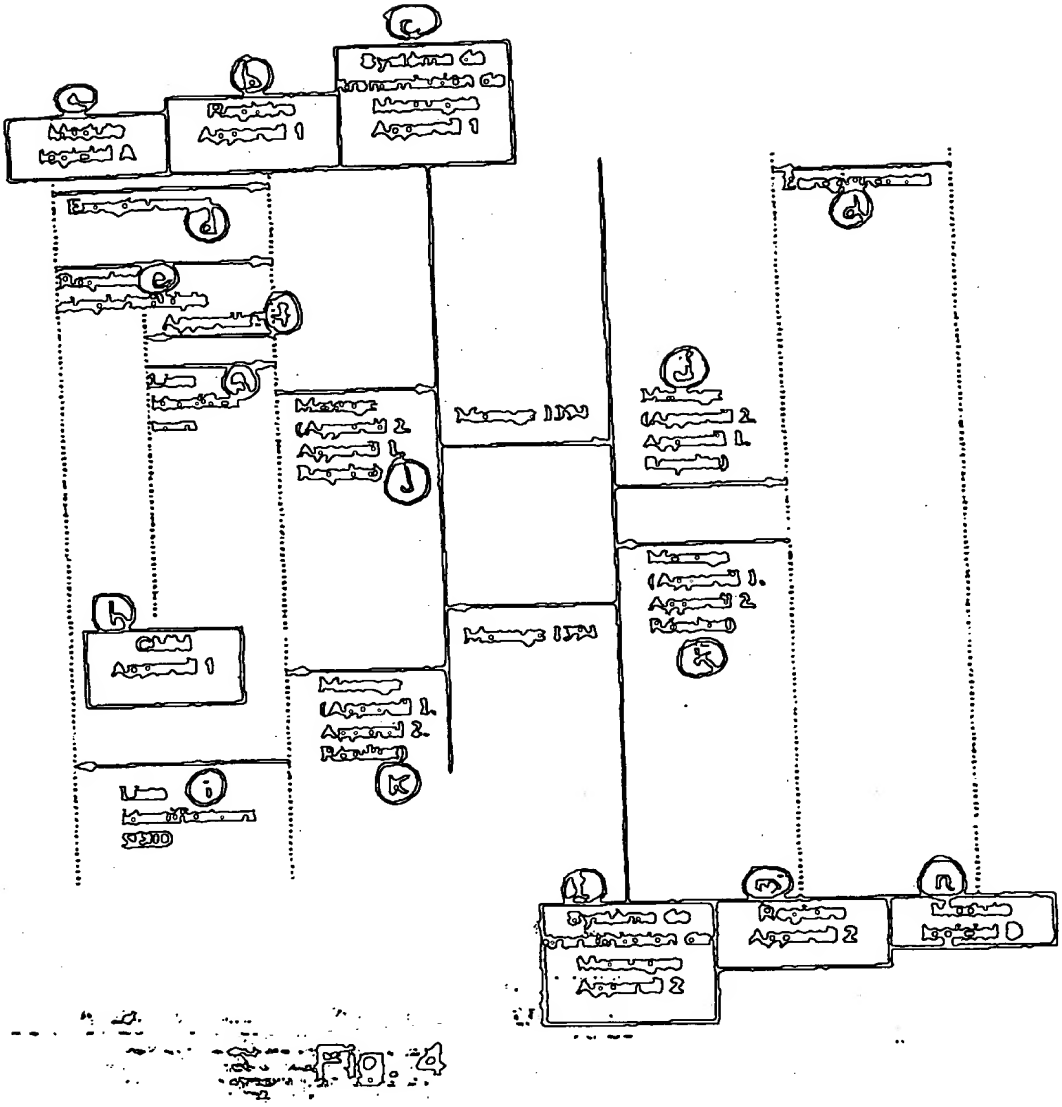

Fig. 6